



# EECS 22: Assignment 3

## Digital Image Processing

Due on Monday 11/04/2013 11:00pm

Note: this is a two-week assignment

# Outline

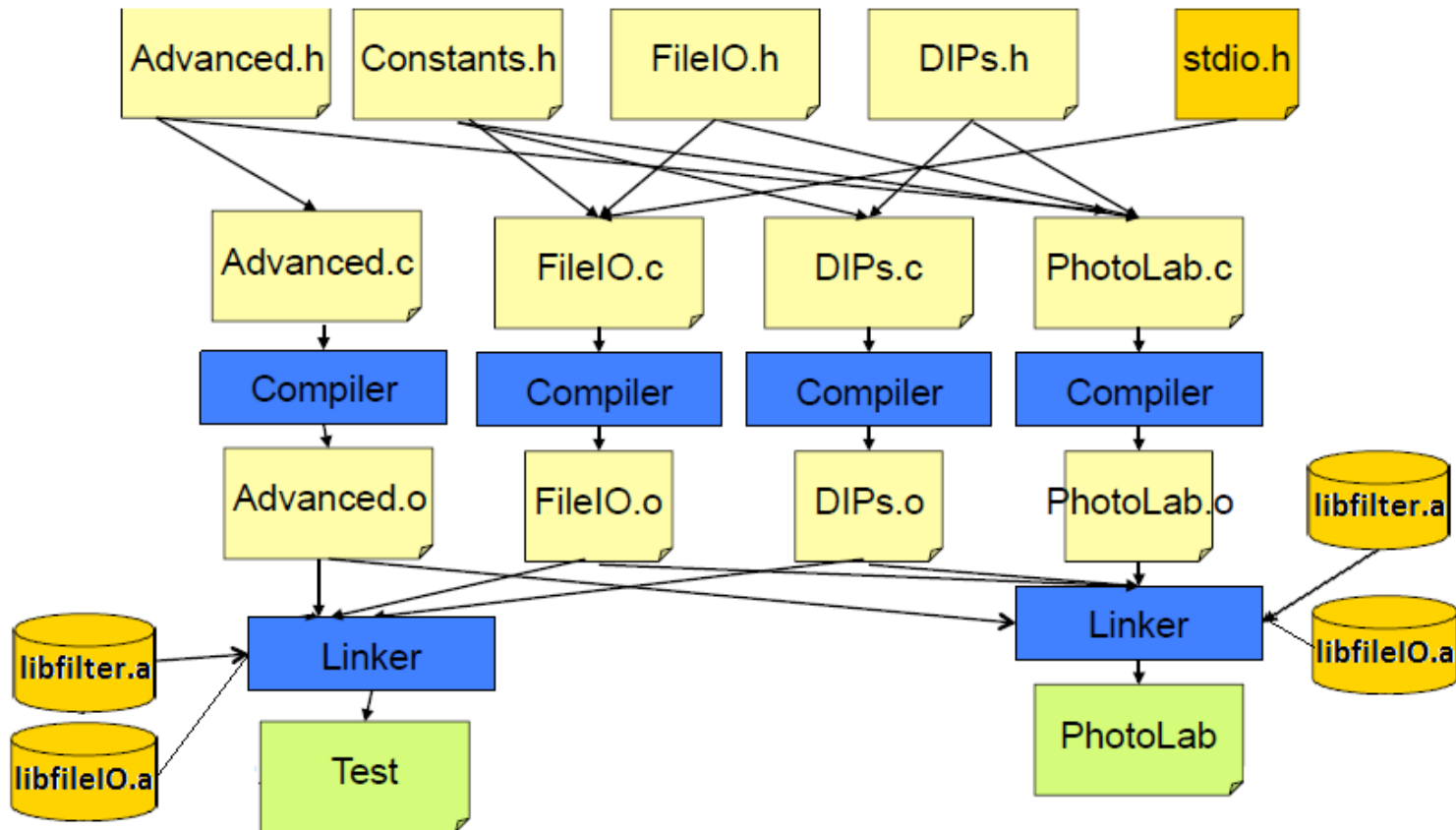
- Decompose a program into multiple modules
- Compile the program with multiple modules using static shared library
- Makefile
- Advanced DIP operation
  - Bit Manipulations: Posterize the image
  - Add Noise to an image
  - Image Overlay
  - Bonus: Surprise DIP Function
- DEBUG mode support
- Extend the Makefile

# Decompose a program into multiple modules

- ❑ **PhotoLab.c:** the main module contains the main() function, and the menu function PrintMenu() as well as AutoTest().
- ❑ **FileIO.c:** the module for the function definitions of ReadImage() and SaveImage().
- ❑ **FileIO.h:** the header file for FileIO.c, with the function declarations of ReadImage() and SaveImage().
- ❑ **Constants.h:** the header file in which the constants to be used are defined.
- ❑ **DIPs.c:** the module for the DIP function definitions in Assignment 2, i.e. BlackNWhite(), VFlip(), HMirror(), ColorFilter(), Sharpen(), Edge().
- ❑ **DIPs.h:** the header file for DIPs.c, with the DIP function declarations.
- ❑ **Advanced.c:** the module for the function definition of new filters in Assignment 3, Posterize(), AddNoise(), and Overlay().
- ❑ **Advanced.h:** the header file for Advanced.c, with the function declarations of Posterize(), AddNoise(), and Overlay().

# Assignment 3

- Compile the program with multiple modules using static shared library



# Compilation

- Compile the program with multiple modules using static shared library

## I. Generate the object files for each module, e.g.

```
% gcc -c FileIO.c -o FileIO.o -ansi -Wall
% gcc -c DIPs.c -o DIPs.o -ansi -Wall
% gcc -c Advanced.c -o Advanced.o -ansi -Wall
% gcc -c PhotoLab.c -o PhotoLab.o -ansi -Wall
```

## II. Create libraries

```
% ar rc libfileIO.a FileIO.o
% ranlib libfileIO.a
% ar rc libfilter.a DIPs.o Advanced.o
% ranlib libfilter.a
```

## III. Linking with the library

```
% gcc PhotoLab.o -lfileIO -lfilter -L. -o PhotoLab
```

## IV. Execute the program

```
% ./PhotoLab
```

# Makefile

- Refer to lecture 8

# DIP operation

## Advanced DIP operation

- Posterize the image
- Add Noise to an image
- Image Overlay
- Bonus:
  - Surprise DIP Function

```
void Posterize(unsigned char R[WIDTH][HEIGHT],  
               unsigned char G[WIDTH][HEIGHT],  
               unsigned char B[WIDTH][HEIGHT],  
               unsigned int rbits,  
               unsigned int gbits,  
               unsigned int bbits);
```

```
void AddNoise( int n,  
               unsigned char R[WIDTH][HEIGHT],  
               unsigned char G[WIDTH][HEIGHT],  
               unsigned char B[WIDTH][HEIGHT]);
```

```
void Overlay( char fname[SLEN],  
             unsigned char R[WIDTH][HEIGHT],  
             unsigned char G[WIDTH][HEIGHT],  
             unsigned char B[WIDTH][HEIGHT],  
             unsigned int x_offset, unsigned int  
             y_off);
```

# Posterize the image

For instance pixel ( 0,0 ) :

$$R[0][0] = 41 = 00101001_2$$

$$G[0][0] = 84 = 01010100_2$$

$$B[0][0] = 163 = 10100011_2$$

- **rbits**, **gbits**, and **bbits** specify the number of least significant bits that need to be posterized. Since the size of unsigned char variable is 8 bits, the valid range of **rbits**, **gbits**, and **bbits** will be 1 to 8.



(a) Posterize the least 6 significant bits of the red channel for pixel(0,0)



(b) Posterize the least 5 significant bits of the green channel for pixel(0,0)



(c) Posterize the least 4 significant bits of the blue channel for pixel(0,0)

Figure 1: The example of posterizing the color channels.



- **HINT:** You will need to use bitwise operators, e.g. '&', '<<', '>>', '|'



# Add Noise

number of noise added to the image:

$$\frac{n * \text{WIDTH} * \text{HEIGHT}}{100}$$

n : Noise percentage



1. Include the `stdlib.h` and `time.h` header files at the beginning of your program:

```
#include <stdlib.h>
#include <time.h>
```

2. Include the following lines at the beginning of your main function:

```
/* initialize the random number generator with the current time */
srand( time(NULL));
```

3. To simulate drawing a card from the shuffled deck, use the following statement:

```
/* generate a random pixel */
x = rand() % WIDTH; // You need to define the variable x.
y = rand() % HEIGHT; // You need to define the variable x.
```

The integer variables **x** and **y** then will have a random values in the range from **WIDTH** and **HEIGHT** accordingly.

# Overlay



(a) Snowboard image

(b) HalloweenI image

(c) HalloweenII image



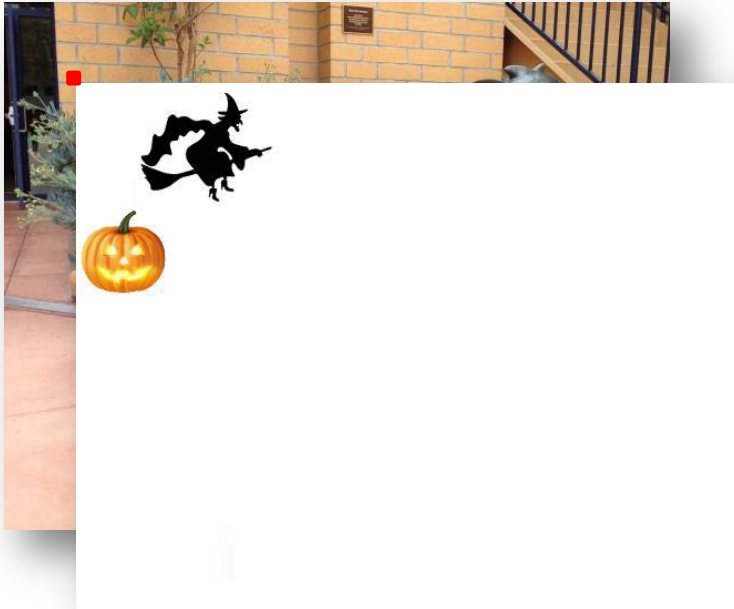
(d) Overlay the SnowBoard image at position (0,0), and the Halloween image at position (50,30)



(e) Overlay the third image at position (70,65)


# Overlay

(70,65)



Each pixel in selected image for overlay operation:

- **white**: RGB value is greater than 250, ( $r \geq 250$ ,  $g \geq 250$ ,  $b \geq 250$ ), this pixel should not be put onto the UCI\_Peter.
- **non white**: overwrite to UCI\_Peter.

- 
- Support for the DEBUG mode
    - Refer to Lecture 9
  - Extend the Makefile
    - Refer to Lecture 8